Automatizirano testiranje email servisa korištenjem Cypress okruženja

Cica, Filip

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:172120

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-21**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

Automatizirano testiranje email servisa korištenjem Cypress okruženja

Diplomski rad

Filip Cica

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za ocjenu diplomskog rada na sveučilišnom diplomskom studiju

Ocjena diplomskog rada na sveučilišnom diplomskom studiju

Ime i prezime pristupnika:	Filip Cica
Studij, smjer:	Sveučilišni diplomski studij Računarstvo
Mat. br. pristupnika, god.	D1272R, 07.10.2022.
JMBAG:	0165083929
Mentor:	doc. dr. sc. Ivan Vidović
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	izv. prof. dr. sc. Tomislav Matić
Član Povjerenstva 1:	doc. dr. sc. Ivan Vidović
Član Povjerenstva 2:	izv. prof. dr. sc. Ivan Aleksi
Naslov diplomskog rada:	Automatizirano testiranje email servisa korištenjem Cypress okruženja
Znanstvena grana diplomskog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	U ovom radu potrebno je istražiti dostupne alate, okruženja i module za automatizirano testiranje email servisa. Naglasak je potrebno staviti na module koji se mogu koristiti u Cypress okruženju. Nakon provedenog istraživanja potrebno je predložiti vlastito rješenje za automatizaciju testiranja email servisa te ga usporediti s postojećim rješenjima. U praktičnom dijelu rada je, osim dizajna i implementacije modula, potrebno primjeniti razvijene module za izradu automatiziranih testova. Tema rezervirana za: Filip Cica Sumentor iz tvrtke (Stefani Majić, Barage

Datum ocjene pismenog dijela diplomskog rada od strane mentora:	08.07.2024.
Ocjena pismenog dijela diplomskog rada od strane mentora:	Izvrstan (5)
Datum obrane diplomskog rada:	15.7.2024.
Ocjena usmenog dijela diplomskog rada (obrane):	Izvrstan (5)
Ukupna ocjena diplomskog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije diplomskog rada čime je pristupnik završio sveučilišni diplomski studij:	15.07.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O IZVORNOSTI RADA

Osijek, 15.07.2024.

Ime i prezime Pristupnika:	Filip Cica
Studij:	Sveučilišni diplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D1272R, 07.10.2022.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: Automatizirano testiranje email servisa korištenjem Cypress okruženja**

izrađen pod vodstvom mentora doc. dr. sc. Ivan Vidović

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD 1
2. PODRUČJE RADA 2
2.1 Email 2
2.1.1 SMTP 2
2.1.2 IMAP 2
2.2. Testiranje email servisa 3
2.2.1. REST sučelje SMTP servera 3
2.2.2. Servis treće strane 5
2.2.3. IMAP protokol 6
3. CYPRESS 7
3.1. Aplikacija 8
3.2. Naredbe 9
3.2.1. Dodavanje naredbe 10
3.3. Izvođenje u node u procesu 10
3.4. Cypress predloženi priključci za testiranje email servisa 12
4. IMAP PRIKLJUČAK ZA CYPRESS 14
4.1. Korisnički zahtjevi 14
4.2. Zahtjevi na programsko rješenje 15
4.2.1. Funkcionalni zahtjevi 16
4.2.2. Nefunkcionalni zahtjevi 19
4.3. Dizajn programskog rješenja 21
4.3.1. Arhitektura 21
4.3.2. Dizajn korisničkog sučelja 22
4.4. Implementacija 24
4.4.1. Odabir tehnologija 24
4.4.2. Funkcije 25
4.4.3. Zadaci 26
4.4.4. Omotači 26
4.5. Korištenje 27
4.6. Automatski testovi 28
4.7. Objavljivanje 28
4.7.1. Dodavanje priključka u cypress dokumentaciju 29
5. USPOREDBA PREDLOŽENOG RJEŠENJA S POSTOJEĆIM 31
6. ZAKLJUČAK 34
LITERATURA 35
ŽIVOTOPIS 37
SAŽETAK 38
ABSTRACT 39
P.4.1 40

1. UVOD

Email servis je ključan dio programskih rješenja poslovnih i mnogih drugih aplikacija. Zadužen je za obavještavanje korisnika i razvojnih inženjera o događajima unutar aplikacije. Dobar primjer je obavještavanje korisnika putem email poruke o autentifikaciji unutar aplikacije što služi kao sigurnosna osobina aplikacije. Osim autentifikacije, sve kupovine, računi, fakture, obavijesti o prijeđenim granicama, magične poveznice i sl. dolaze putem email poruke. Jedna aplikacija može sadržavati veliki broj email poruka koje je nužno testirati. Manualno testiranje ovakvih servisa je vrlo naporan i dugotrajan proces jer uključuje aktiviranje email poruke akcijama nad sučeljem aplikacije stoga je ovakve testove nužno automatizirati.

Cypress je alat za automatsko testiranje u pregledniku i ne dolazi sa sustavom za automatsko testiranje email poruka, već se oslanja na priključke od zajednice. Priključci od zajednice se uglavnom svode na adaptere za REST ili programska sučelja vlastitih email servera ili servisa treće strane koji nude testne email servere. Navedeni pristupi rade vrlo dobro, no imaju nekoliko mana: nemoguće ih je testirat naspram postojećih email servera kao što je gmail, potrebno je instalirati vlastiti email server s REST sučeljem za povlačenje poruka i u slučaju servisa treće strane potrebno je kupiti ključ.

Ovim radom predloženo je rješenje koje koristi IMAP kao centralnu tehnologiju interakcije s email serverom i rješava prethodno navedene probleme. Korištenjem IMAP protokola moguće je validirati funkcionalnost razvijenog email servisa zaduženog za slanje email poruka korisnicima aplikacije s postojećim email serverima kao što su gmail i outlook. Nadalje, nije potrebno instalirati vlastiti server već je moguće koristiti postojeći server kao što je gmail ili bilo koji drugi. Ako vlastiti server već postoji, moguće je koristiti IMAP i s njime jer se radi o standardiziranom protokolu.

Nakon uvodnog poglavlja napravljen je pregled područja rada gdje je ukratko objašnjeno što je email i na kojim protokolima se temelji te je napravljan pregled postojećih tehnika testiranja s primjerima. U trećem poglavlju napravljen je pregled cypress razvojnog okruženja te su na kraju poglavlja objašnjena postojeća rješenja za testiranje email servisa u cypressu. Četvrto poglavlje prikazuje postupak definiranja zahtjeva, dizajna, implementacije te testiranje i objavljivanje priključka predloženog u sklopu ovog diplomskog rada. U zadnjem poglavlju napravljena je usporedba karakteristika postojećih rješenja s rješenjem predloženim u sklopu ovog rada.

2. PODRUČJE RADA

U ovom poglavlju naveden je pregled područja rada koji uključuje osnove o email tehnologiji, email servisima te pregled postojećih pristupa testiranju email servisa. Pristupi su objašnjeni neovisno o tehnologiji tj. mogu biti implementirani bilo kojim programskim jezikom u bilo koje okruženje.

2.1 Email

Sedamdesetih godina 20. stoljeća implementirana je prva decentralizirana računalna mreža ARPANET (engl. *Advanced Research Projects Agency Network*) na kojoj dvojica programera Beranet i Newman experimentiraju sa slanjem poruka [1]. To experimentiranje sa slanjem poruka početak je elektroničkih poruka (engl. *electronic mail, e-mail, email*). Danas se za elektroničku poruku (e-poruka), elektroničku adresu (e-adresa) i zajedno s drugim povezanim pojmovima (npr. email tehnologija) koristi anglicizam email.

Kroz vrijeme, email tehnologija se razvijala kroz nizove inovacija te je bez nje internet nezamisliv. Za pristup većini usluga potrebno je navesti email adresu na koju često stižu email poruke potvrde, email poruke sa sigurnosnim kodovima, reklamne email poruke i razne ostale.

U pravilu email poruke se distribuiraju pomoću email servera. Email server je računalo koje omogućava primanje i slanje email poruka. To radi pomoću protokola poput: SMTP, IMAP i POP3. U ovom poglavlju su ukratko objašnjeni prethodno navedeni protokoli na kojima se zasniva moderna email tehnologija.

2.1.1 SMTP

SMTP (engl. *Simple Mail Transfer Protocol*) je protokol čija je svrha raspodjeljivanje email poruka s jednog email servera prema jednom ili više drugih email servera na internetu. SMTP protokol otvara TCP (engl. *Transfer Control Protocol*) vezu između pošiljatelja i primatelja te šalje poruku sa SMTP pošiljatelja ka SMTP primatelju tj. primateljima. Poruka je poslana serijom zahtjev-odziv transakcija između sudionika i sastoji se od zaglavlja i tijela gdje se nalaze podaci o poruci, pošiljatelju i primatelju [2].

2.1.2 IMAP

IMAP (engl. *Internet Message Access Protocol*) je protokol čija je svrha pristup porukama u sandučiću (engl. *mailbox*) koji se nalazi na udaljenom serveru. S IMAP klijentom moguće je:

znati kad je nova poruka došla, dijeljenje sandučića s bilo kime, prebacivanje poruka između sandučića i označavanje poruka zastavicama [3].

Osim IMAP protokola postoji još jedan dobro poznat protokol za pristup porukama pod imenom POP (engl. *Post Office Protocol*). POP pruža sličnu funkcionalnost kao i IMAP uz nekoliko ključnih razlika. POP je jednostavniji i time brži od IMAP protokola dok je IMAP protokol kompliciraniji, ali nudi više funkcionalnosti [4]. IMAP protokol troši više resursa ali, i omogućava pristup sandučiću s više uređaja istovremeno dok POP protokol to ne omogućava [4]. Za automatsko testiranje pristup sandučiću s više uređaja je nužna funkcionalnost.

2.2. Testiranje email servisa

Email servis je dio aplikacije koji je zadužen za slanje email poruka na temelju učinjenih akcija u aplikaciji. Testiranje email servisa odnosi se na verifikaciju i validaciju rada takvog servisa. Klasični primjer slučaja kojeg je potrebno testirati je autentifikacija u aplikaciju. Nakon autentifikacije korisnik dobiva obavijest o autentifikaciji putem email poruke s raznim detaljima. Potrebno je testirati: primitak poruke, naslov poruke, sadržaj poruke i funkcionalnosti unutar poruke. Funkcionalnosti unutar poruke odnose se na HTML (engl. *Hyper Text Markup Language*) koji sadrži poveznice [5]. Klikom na poveznicu se okida akcija na serverskoj strani aplikacije. Osim navedenog postoji još veliki broj slučajeva gdje se email poruka treba provjeriti. Također, jedna aplikacija može sadržavati desetke različitih email poruka od kojih je svaka drugačija i svaka se okida različitom akcijom unutar aplikacije. Sve te email poruke je potrebno testirati. Ručno testiranje email servisa može biti jako naporan i mukotrpan posao. Zbog čega je dobro automatizirati takve testove.

Trenutno stanje tehnologija za automatsko testiranje email servisa općenito se svodi na implementaciju vanjskih programskih rješenja koja omogućavaju pristupanje sadržaju sandučića kako bi se potvrdio primitak email poruke i njezin sadržaj. Postoji nekoliko načina pristupu email poruka unutar sandučića: pomoću SMTP servera koji ima REST[1] sučelje, pomoću servisa treće strane. pomoću IMAP protokola.

2.2.1. REST sučelje SMTP servera

Testiranje email servisa pomoću SMTP servera se može ostvariti: instaliranjem servera na lokalno računalo, instaliranjem servera na internet, ili korištenjem postojećeg servera. Kod ovog

[1] "REST je koordinirani skup arhitekturalnih ograničenja koji se pokušava svesti na najmanju moguću mjeru latencije i mrežne komunikacije uz istodobno maksimiziranje neovisnosti i skalabilnost implementacija komponenti.", R.T., Fielding, **Arhitekturalni stilovi i dizajn mrežnih arhitektura baziranih na softveru**: doktorska disertacija, University of California, Irvine, str. 148, 2000.

pristupa ostvaruje se ovisnost o REST sučelju koje SMTP server pruža tj. u slučaju da je SMTP server u kvaru automatski testovi ne rade sve dok server nije popravljen.

Instaliranje servera na lokalno računalo podrazumijeva ne postojanje razvojnog okruženja za testiranje gdje postoji posebna grana aplikacije specifično namijenjena za testiranje koja je uobičajeno instalirana unutar zaštićene mreže. U ovom slučaju testni inženjer lokalno instalira aplikaciju i SMTP server te piše i izvodi automatske testove [6].

Instaliranje servera na internet podrazumijeva postojanje testnog okruženja gdje je aplikacija podignuta na internet. U ovom slučaju potrebno je instalirati SMTP server. Instaliranje SMTP servera je kompleksan proces koji uključuje postavljanje email DNS zapisa, sigurnosnog sloja (engl. *Transport Layer Security, TLS*), instaliranje servera te konfiguraciju servera [7].

Primjer SMTP servera koji se može koristiti lokalno ili instalirati na internet je maildev[2]. Maildev je alat koji pruža SMTP server za testiranje, dolazi s grafičkim sučeljem i REST sučeljem [8].

```
const MailDev = require("maildev");

const maildev = new MailDev();

maildev.listen();

maildev.on("new", function (email) {
  // We got a new email!
});
```

Slika 2.1. Primjer povlačenja pošte s maildev SMTP poslužitelja[3].

Nakon što je maildev podignut lokalno ili na internet, pošti je moguće pristupiti programski kao što je prikazano na slici 2.1. Alat sličan maildevu koji se također često koristi za izradu automatskih testova je mailhog[4] koji je detaljno objašnjen u poglavlju 3.4.

Za korištenje postojećeg servera nije bitno testno okruženje. Koristi se postojeći SMTP server poput gmail servera. Gmail omogućuje povlačenje email poruka preko REST sučelja. Kako bi se ova funkcionalnost omogućila potrebno je stvoriti projekt na Google API konzoli, omogućiti Gmail API, omogućiti autentifikaciju te odabrati jedan od ponuđenih jezika te početi s implementacijom [9]. Dio gmail API implementacije prikazan je na slici 2.2.

[2] Maildev, https://github.com/maildev/maildev, 05.06.2024.
[3] Maildev API, https://github.com/maildev/maildev?tab=readme-ov-file#api, 19.06.2024.
[4] Mailhog, https://github.com/mailhog/MailHog, 05.06.2024.

```
var GoogleAPIMailClient = window.GoogleAPIMailClient || (function() {

var clientId,
    apiKey,
    scopes = 'https://www.googleapis.com/auth/gmail.readonly',
    Messages = {};

function config(config) {
  clientId = config.clientId;
  apiKey = config.apiKey;
}

function clientLoad() {
  gapi.client.setApiKey(apiKey);
  window.setTimeout(checkAuth, 1);
}
...
```

Slika 2.2. Korištenje gmail REST sučelja[5].

2.2.2. Servis treće strane

Koristi se servis treće strane koji omogućava stvaranje jednog ili više sandučića i pristup email porukama unutar sandučića programskim putem. Ovakvi servisi uvijek naplaćuju svoju uslugu. Implementacijom trećeg servisa također se ostvaruje ovisnost o sučelju tog servisa.

Primjer ovakvog pristupa postiže se servisom kao što je mailosaur[6]. Mailosaur je servis koji omogućava automaciju email testova i SMS poruka te također daje pristup SMTP pseudo serverima (engl. *dummy server*) [10].

```
...
constructor() {
  const defaultApiKey = Cypress.env('MAILOSAUR_API_KEY');
  this.mailosaurSetApiKey(defaultApiKey);
}

mailosaurSetApiKey(apiKey) {
  this.request = new Request({ apiKey, baseUrl: Cypress.env('MAILOSAUR_BASE_URL')
});

mailosaurListServers() {
  return this.request.get('api/servers');
}
...
```

Slika 2.3. Primjer korištenja mailosaur servisa u javascriptu[7].

[5] Gmail-api-js main datoteka, https://github.com/ademidoff/gmail-api-js/blob/master/main.js, 19.06.2024.
[6] Mailosaur, https://mailosaur.com/, 05.06.2024.
[7] Cypress-mailosaur-naredbe, https://github.com/mailosaur/cypress-mailosaur/blob/main/src/mailosaurCommands.js, 19.06.2024.

Na slici 2.3. prikazan je primjer korištenja mailosaur servisa gdje su prikazani konstruktor koji koristi metodu za postavljanje API ključa mailosaur servisa i metoda za izlistavanje pseudo servera. Osim mailosaura sličan servis je mailslurp[8] a postoji i besplatni servis guerrillamail[9] koji pruža besplatne sandučiće za testiranje.

2.2.3. IMAP protokol

IMAP protokolom se povlače email poruke iz sandučića u testni kod. Potrebno je imat sandučić na bilo kojem SMTP serveru te je potrebno na SMTP serveru omogućiti IMAP protokol ako već nije omogućen. Kod ovog pristupa ne stvara se ovisnost o REST sučelju SMTP servera. Stvara se ovisnost o IMAP protokolu što je povoljno jer nije potrebno instalirati vlastiti SMTP server, a ako server koji se koristi prekine djelovati, moguće je napraviti sandučić na bilo kojem drugom serveru i nastaviti s radom. Ovakav pristup je besplatan.

[8]Mailslurp, https://www.mailslurp.com/, 05.06.2024.
[9]Guerrilamail, https://www.guerrillamail.com/, 05.06.2024.

3. CYPRESS

Cypress je okruženje za testiranje web aplikacija. Dvije glavne vrste testiranja koje cypress pruža su e2e (engl. *end to end*) testiranje i testiranje komponenti. E2e testiranje je korištenje programskog rješenja isto kako bi ga i korisnik koristio. E2e testiranjem se pokriva opsežan dio automatske verifikacije i validacije operativnosti programskog rješenja te ovakav pristup zahtjeva punu funkcionalnost serverskog i klijentskog dijela aplikacije [11]. Testiranje komponenti je simuliranje stanja izolirane komponente kako bi se provjerila funkcionalnost te komponente [11]. Osim navedenog cypress omogućuje unit i integracijsko testiranje tj. bilo kakvo testiranje unutar preglednika kojeg cypress instancira u node procesu. Node proces i cypress kontinuirano komuniciraju kroz izvođenje, sinkroniziraju se i izvršavaju zadatke jedno za drugo [12]. Testovi se izvode manipulacijom DOM[10] (engl. *Document Object Model*) sučelja unutar instanciranog preglednika pomoću javascript koda koji dolazi u obliku cypress naredbi. Cypress također omogućuje upravljanje mrežom tj. omogućuje presretanje poziva i mijenjanje sadržaja kroz izvođenje [12]. Cypress implicitno čeka pojavu elementa nad kojim treba izvršiti operacije tj. uklanja potrebu za explicitnim čekanjima unutar testnog koda. Što se tiče pronalaženja grešaka (engl. *debugging*) cypress pruža stotine poruka greške (engl. *error messages*), grafičko sučelje koje prikazuje izvođenje, uzimanje slika (engl. *snapshot*) kroz izvođenje i ispisivanje detaljnih podataka o grešci u konzolu razvojnih alata (engl. *developer tools console*) [13].

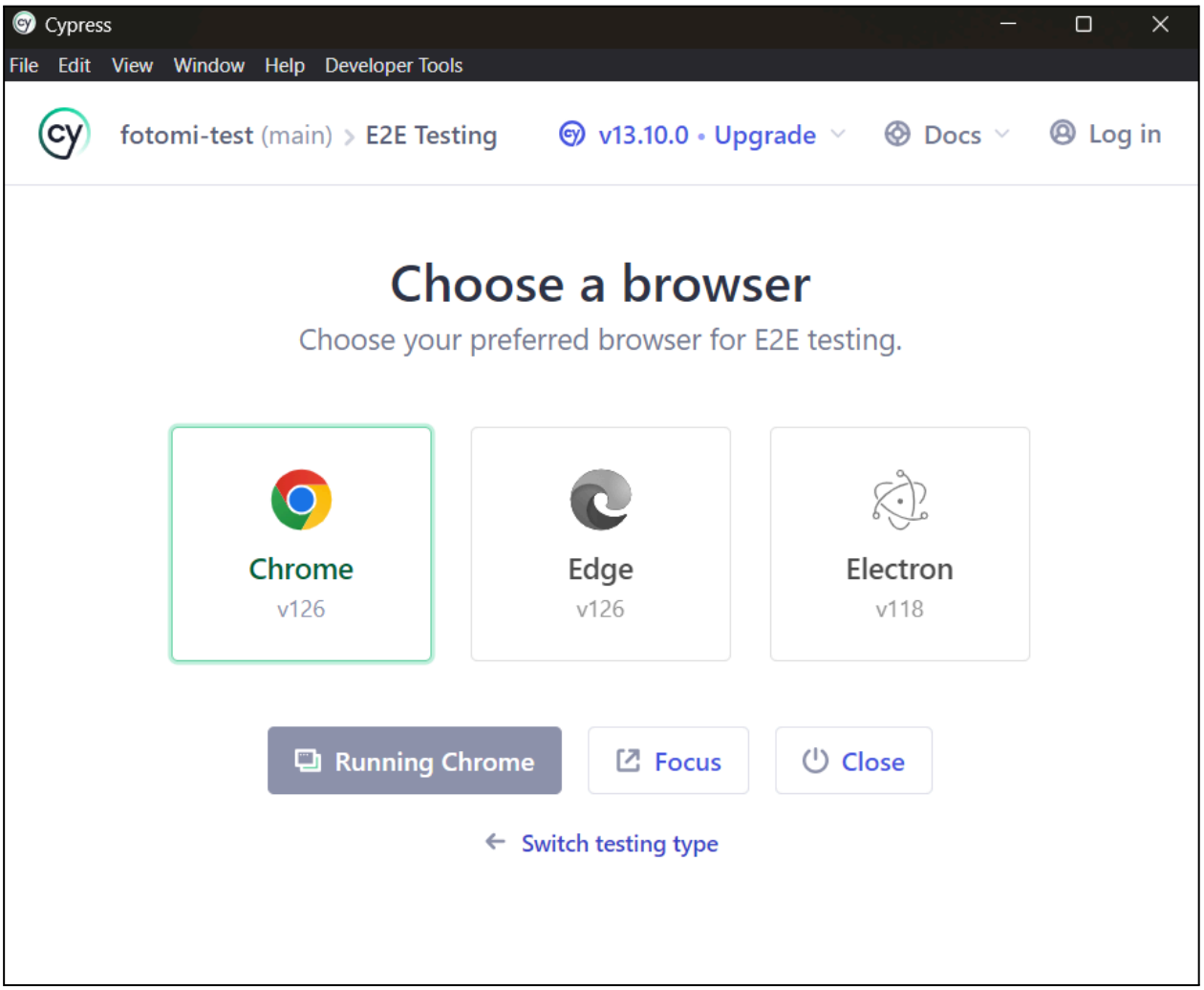
Cypress podržava nekoliko preglednika: Chrome, Edge, Firefox i Electron te dolazi s aplikacijom, robustnim programskim sučeljem, raznim dodatnim alatima, sustavom za dodavanje vlastitih naredbi i sustavom za dodavanje priključaka.

Cypress nije sveopći alat za korištenje. Radi jednu stvar, a to je testiranje unutar preglednika. Naredbe se izvode unutar preglednika što znači da je izvođenje metoda koje zahtijevaju node proces otežano. Moguće je pomoću *task* metode izvoditi zadatke u node procesu i vratiti rezultat u cypress te ga tamo provjeriti [14]. Nadalje, nije moguće otvoriti dva preglednika odjednom što može otežati automatiziranje nekih testnih slučajeve. I na kraju, svaki test u cypresu je zalijepljen za jednu super domenu što otežava prebacivanje s jedne super domene na drugu. Ako je takvo ponašanje nužno može se koristiti *origin* metoda iz cypress programskog sučelja.

[10] "DOM je programsko sučelje za validne HTML i dobro formirane XML dokumente", A., Le Hors, P., Le Hegaret, G., Nicol, L., Wood, M., Champion, S.,Byrne **DOM Razina 3 Core Specifikacija**: Verzija 1., str. 13., 2002.

3.1. Aplikacija

Cypress aplikacija je grafičko korisničko sučelje za pokretanje i izvođenje testova te pronalaženje grešaka unutar testova.

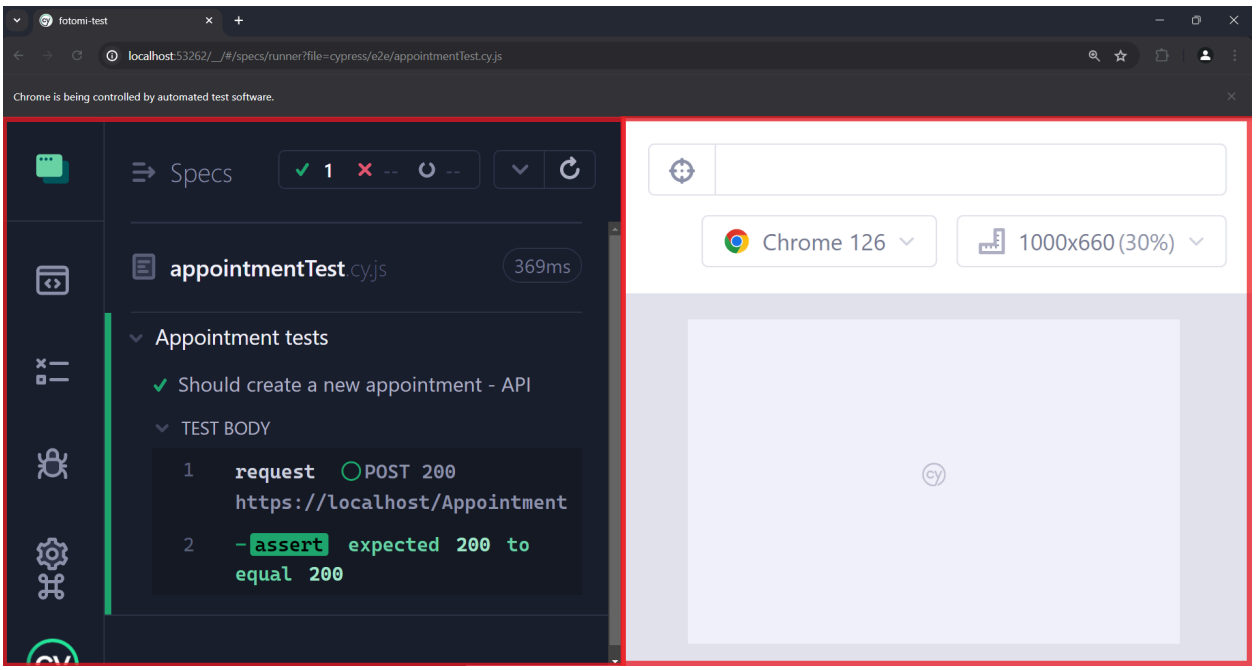


Slika 3.1. Desktop dio cypress aplikacije.

Aplikacija se sastoji od desktop aplikacije (Slika 3.1.) na kojoj se odabire vrsta testiranja i preglednik. Nakon što je preglednik odabran pokreće se web dio aplikacije.

Web dio aplikacije otvara ekran gdje se bira skupina testova koja se izvršava i nakon toga se otvara ekran gdje se testovi izvode (Slika 3.2.). Web aplikacija se pokreće na lokalnom poslužitelju (engl. *localhost*). Sastoji se od dijela za prikazivanje naredbi koje se izvode (Slika 3.2. lijeva strana) i dijela gdje se aplikacija pod testom prikazuje (Slika 3.2. desna strana). Ako aplikacija pod testom nema korisničko sučelje desni dio ekrana je prazan. Ekran na kojemu se prikazuje izvođenje testova radi na način da nakon izvođenja svake naredbe uzima sliku DOM sučelja. Tijekom izvođenja testa svaka naredba koja je pozvana prikazuje se s desne strane.

Nakon izvođenja moguće je prijeći mišem iznad naredbe i zamrznuti sliku prije ili poslije izvođenja naredbe.



Slika 3.2. Web dio aplikacije.

3.2. Naredbe

U trenutku kad je pokrenuta, naredba se ne izvršava, već se dodaje u red naredbi gdje se naredbe izvršavaju jedna poslije druge. Naredbe je također moguće ulančati jer nakon izvođenja naredbe ona vraća subjekt na kojemu je opet moguće pozvati neku naredbu. Cypress dolazi sa širokim skupom naredbi za razne svrhe. Naredbe se mogu podijeliti na: upite, provjere (engl. *assertions*), akcije i dodatne naredbe.

Upiti su naredbe koje čitaju stanje nekog elementa aplikacije pod testom. Najčešća naredba ovog tipa je naredba *get.* Naredba *get* prima selektor nekog elementa i vraća subjekt nad kojim je moguće pozvati druge naredbe.

Provjere su naredbe koje provjeravaju stanje aplikacije pod testom. Najčešća naredba ovog tipa je naredba *should.* Naredba *should* se često ulančava na naredbu *get* i omogućava izvršavanje provjere nad elementom kojeg naredba *get* vrati.

Akcije su naredbe koje omogućuju radnje nad elementima koje bi korisnik napravio. Dobar primjer akcije je naredba *click* koja omogućuje simuliranje klika nad elementom. Osim navedenih postoje mnoge druge naredbe koje imaju razne svrhe kao sto su: manipuliranje kolačićima i pohranom, manipulacija vremenom, presretanje zahtijeva i sl.

Osim naredbi cypress pruža dodatne funkcionalnosti preko globalnog Cypress objekta. Neke od dodatnih funkcionalnosti su metode za pristup podacima o okruženju, metode za modificiranje funkcionalnosti, metode za dodavanje funkcionalnosti i sl.

3.2.1. Dodavanje naredbe

Cypress dolazi sa sučeljem za dodavanje naredbi [15]. Dodavanje naredbi služi za apstrahiranje ponavljajućih procedura kako bi se modifikacija tih procedura radila na jednom mjestu. Dodavanje naredbi umanjuje duplikaciju koda i pridonosi čitljivosti. Najčešća naredba koja se dodaje je naredba za autentifikaciju (Slika 3.3.).

```
import shopPage from "../pages/shopPage"
Cypress.Commands.add("logInThroughUI", (username, password)=>{
    shopPage.logInButton().click()
    shopPage.loginModal.usernameInput().type(username)
    shopPage.loginModal.passwordInput().type(password)
    shopPage.loginModal.loginButton().click()
})
```

Slika 3.3. Dodavanje naredbe za login.

Prema Slici 3.3. dodavanje naredbe se izvršava pozivanjem *add* metode na *Commands* objektu kojemu se pristupa preko Cypress objekta. Naredbu je potrebno dodati prije izvođenja testova gdje se koristi. Dobra praksa je pozive *add* metode smjestiti u odvojenu javascript datoteku te ju uvesti u *e2e.js* koja se izvodi prije testova i time dodaje sve korisničke naredbe u cypress. Naredbe dodane na ovaj način moguće je ulančati s bilo kojom drugom cypress naredbom.

3.3. Izvođenje u node u procesu

Cypress dolazi sa sučeljem za dodavanje priključaka. Priključci mogu biti vlastite funkcije koje se izvode u node procesu ili funkcije iz uvezenih priključaka koje je potrebno koristiti unutar cypress testova

```
const { defineConfig } = require("cypress");

module.exports = defineConfig({
  e2e: {
    setupNodeEvents(on, config) {
      on('task', {
        consoleLog(data) {
          console.log(data);
          return null;
        },
      });
    },
  },
});
```

Slika 3.4. Datoteka cypress.config.js.

Priključci se dodaju u unutar *cypress.config.js* datoteke unutar *setupNodeEvents* metode (Slika 3.4.). Metoda prima dva parametra *on* i *config*. *On* je funkcija koja omogućava postavljanje slušatelja (engl. *listener*) na neki događaj. *Config* je objekt koji sadrži razne vrijednosti koje mogu biti od koristi kod dodavanja slušatelja. *SetupNodeEvents* se poziva kad se cypress otvori te stvara node process unutar projekta koji ima pristup svim priključcima unutar *node_modules* direktorija.

```
Cypress.Commands.add("consoleLog", (data) =>
{
  cy.task("consoleLog", data);
});
```

Slika 3.5. Datoteka commands.js.

Nakon što je metoda dodana moguće ju je koristiti preko *cy.task* metode te je dobra praksa omotati poziv unutar korisničke metode (Slika 3.5.) kako bi ju bilo lakše koristiti.

```
describe('Example Test', () => {
  it('should log Hello World', () => {
    cy.consoleLog("Hello World");
  });
});
```

Slika 3.6. Datoteka test.js.

Korištenje zadatka preko omotača unutar testa prikazano je na slici 3.6.

3.4. Cypress predloženi priključci za testiranje email servisa

Cypress preporuča neke priključke razvijene od zajednice. Priključci predloženi od cypressa za testiranje email servisa su: cypress-mailosaur[11], cypress-maildev[12], cypress-mailslurp[13], cypress-mailhog[14], cypress-guerillamail[15] i cypress-temp-mail[16]. U ovom poglavlju je napravljen pregled tih priključaka.

Cypress-mailosaur je priključak koji pruža programsko sučelje za korištenje mailosaur servisa unutar cypresa. Ovaj priključak radi na način da koristi mailosaurovo REST sučelje kako bi izvršavao akcije s njihovim servisom. Za korištenje servisa potrebno je kupiti pristup servisu te generirati API ključ. Dio implementacije ovog priključka je prikazan na slici 2.3.

Cypress-maildev je priključak koji pruža programsko sučelje za korištenje maildev alata unutar cypressa. Cypress-maildev interno koristi maildev REST sučelje za interakcije s email sandučićem. Maildev je alat otvorenog koda (engl. *opensource*) i besplatan je za korištenje.

Cypress-mailslurp je priključak koji služi kao adapter za korištenje mailslurp-client priključka u cypressu. Mailslurp-client je priključak koji pruža programsko sučelje za korištenje mailslurp servisa. Mailslurp je servis koji omogućava automaciju email testova i SMS poruka. Mailslurp-client interno koristi mailslurp REST sučelje za interakciju s email sandučićem. Mailslurp-client je alat otvorenog koda dok se mailslurp kao servis plaća te je potrebno generirati API ključ za korištenje.

Cypress-mailhog je priključak koji pruža programsko sučelje za interakciju s mailhog serverom. Mailhog je alat za testiranje email servisa koji omogućava konfiguriranje i objavljivanje SMTP servera, dolazi s grafičkim sučeljem i pruža REST sučelje za dohvaćanje email poruka [16]. Cypress-mailhog koristi mailhog REST sučelje za interakciju sa serverom. Cypress-mailhog i mailhog su alati otvorenog koda i nikakvo plaćanje nije potrebno za njihovo korištenje.

Cypress-guerillamail je priključak koji pruža programsko sučelje za stvaranje privremenog sandučića te povlačenja email poruka s istog. Guerillamail je web stranica gdje je moguće stvoriti privremeni sandučić i koristiti ga. Cypress-guerillamail funkcionira preko grafičkog sučelja tj. putem cypress posjećuje stranicu koja automatski stvara novi sandučić i očitava

[11] Cypress-mailosaur, https://github.com/mailosaur/cypress-mailosaur, 05.06.2024.
[12] Cypress-maildev, https://github.com/Clebiez/cypress-maildev, 05.06.2024.
[13] Cypress-mailslurp, https://github.com/mailslurp/cypress-mailslurp, 05.06.2024.
[14] Cypress-mailhog, https://github.com/SMenigat/cypress-mailhog, 05.06.2024.
[15] Cypress-guerillamail, https://github.com/e23thr/cypress-guerrillamail, 05.06.2024.
[16] Cypress-temp-mail, https://github.com/madhusaran/cypress-temp-mail, 19.06.2024.

podatke email poruke direktno iz preglednika. Guerrillamail je potpuno besplatna usluga otvorenog koda.

4. IMAP PRIKLJUČAK ZA CYPRESS

U sklopu ovog diplomskog rada predloženo je programsko rješenje za testiranje email servisa pomoću IMAP protokola u obliku cypress priključka. Trenutno ne postoji takav priključak. Priključak je moguće koristiti i izvan cypressa u bilo kojem projektu koji koristi javascript zbog jezične kompatibilnosti. Priključak se zove cymap. Kroz ovo poglavlje prikazan je postupak razvoja tog priključka.

4.1. Korisnički zahtjevi

Korisnički zahtjevi predstavljaju potrebe, očekivanja i ograničenja korisnika ili dionika koji se susreću s programskim rješenjem [17]. Prije definiranja zahtjeva potrebno je navesti dionike (engl. *stakeholders*). Dionici uključuju osobe, grupe ili organizacije koje sudjeluju u projektu, koje ishod projekta pogađa ili koje mogu utjecati na ishod projekta [18]. Projekt u sklopu ovog rada je izrada cypress priključka. Popis dionika zajedno s njihovim karakteristikama prikazan je na tablici 4.1.

Tablica 4.1. Popis i karakteristike dionika.

Dionik	**Tip**	**Opis**	**Odnos s projektom**
Razvojni inženjer vlasnik	Osoba	Glavni inženjer koji radi na razvoju priključka	Aktivno uključen u projekt
Korisnik	Osoba	Razvojni inženjer testne automacije	Pogođen ishodom

Za svakog od dionika su definirani korisnički zahtjevi. Za definiranje korisničkih zahtjeva su korištene korisničke priče. Korisnička priča je sažeta izjava koja objašnjava korisnikove potrebe i služi kao početna točka za definiranje detalja [19]. Korisničke priče za svakog od dionika su navedene u tablicama 4.2. i 4.3.

Tablica 4.2. Korisničke priče korisnika priključka.

R.b.	Korisnička priča
1.	Kao korisnik želim imati mogućnost pristupanja sadržaju poštanskog sandučića programski
2.	Kao korisnik želim imati mogućnost brisanja sadržaja poštanskog sandučića programski
3.	Kao korisnik želim jednostavno i intuitivno programsko sučelje
4.	Kao korisnik želim da priključak ispisuje smislene i opisne greške
5.	Kao korisnik želim da priključak sadrži dokumentaciju
6.	Kao korisnik želim da priključak izvodi operacije u razumnom vremenu
7.	Kao korisnik želim da priključak bude kompatibilan s najnovijom verzijom cypressa
8.	Kao korisnik želim da priključak ima minimalan broj ovisnosti o drugim priključcima

Tablica 4.3. Korisničke priče razvojnog inženjera priključka.

R.b.	Korisnička priča
1.	Kao razvojni inženjer želim da priključak rukuje greškama radi lakšeg pronalaženja problema
2.	Kao razvojni inženjer želim imati automatske testove za funkcionalnosti priključka
3.	Kao razvojni inženjer želim da priključak prati neku konvenciju imenovanja
4.	Kao razvojni inženjer želim da priključak bude proširiv

4.2. Zahtjevi na programsko rješenje

Definiranje zahtjeva važan je korak u razvoju programskih rješenja. Za programsko rješenje predloženo ovim radom definirani su funkcionalni i nefunkcionalni zahtjevi. Analiza funkcionalnih i nefunkcionalnih zahtjeva uključuje razumijevanje svakog zahtjeva i predstavljanje skupina tih zahtjeva na više različitih načina [19]. Funkcionalni i nefunkcionalni zahtjevi su definirani na temelju korisničkih zahtjeva navedenih u prethodnom poglavlju.

4.2.1. Funkcionalni zahtjevi

Za definiranje funkcionalnih zahtjeva priključka korištena je metoda definiranja slučajeva korištenja. Slučaj korištenja (engl. *use case*) je tekstualni opis ponašanja sustava pod različitim uvjetima dok obavlja funkciju koju je dionik pokrenuo [18].

Tablica 4.7. Slučaj korištenja: Uspostava veze.

Slučaj korištenja	Uspostavi vezu s email serverom
Preduvjet	Korisnik je omogućio IMAP na svome email serveru
Opis	
Korak	**Akcija**
1.	Korisnik poziva metodu za uspostavljanje veze.
2.	Korisnik postavlja lozinku i korisničko ime sandučića na koji se želi povezati u detalje za povezivanje
3.	Korisnik postavlja poslužitelja u detalje za povezivanje
4.	Korisnika postavlja port na 993 i tls na true u detalje za povezivanje
5.	Korisnik pokreće kod
6.	Veza se uspostavlja
Extenzije	
Korak	**Akcija**
2a	Lozinka je neispravna
2b	Korisnik je neispravan
3a	Poslužitelj je neispravan
4a	Port je neispravan

U tablici 4.7. naveden je slučaj korištenja koji opisuje funkcionalnost uspostavljanja veze s email serverom. Slučaj korištenja je napisan na temelju korisničkih priča 1 i 2 iz tablice 4.2.

Tablica 4.8. Slučaj korištenja: Povlačenje pošte.

Slučaj korištenja	Povuci svu poštu s email servera
Preduvjet	Metoda za povezivanje sa serverom je pozvana
Opis	
Korak	**Akcija**
1.	Korisnik poziva metodu za povlačenje pošte s email servera
2.	Korisnik pokreće kod
3.	Email pošta je uspješno povučena u programski kod
Extenzije	
Korak	**Akcija**
3a	Sandučić je prazan

U tablici 4.8. naveden je slučaj korištenja koji opisuje funkcionalnost povlačenja email pošte u programski kod. Slučaj korištenja je napisan na temelju korisničke priče 1 iz tablice 4.2.

Tablica 4.9. Slučaj korištenja: Povlačenje pošte po indeksu.

Slučaj korištenja	Povuci email poruku s email servera prema navedenom indeksu poruke
Preduvjet	Veza sa serverom je uspostavljena
Opis	
Korak	**Akcija**
1.	Korisnik poziva metodu za povlačenje pošte s email servera
2.	Korisnik navodi indeks
3.	Korisnik pokreće kod
4.	Email poruka je uspješno povučena u testni kod
Extenzije	
Korak	**Akcija**
2a	Index je negativan
4a	Sandučić je prazan

U tablici 4.9. naveden je slučaj korištenja koji opisuje funkcionalnost povlačenja email pošte u programski kod. Slučaj korištenja je napisan na temelju korisničke priče 1 iz tablice 4.2.

Tablica 4.10. Slučaj korištenja: Brisanje pošte.

Slučaj korištenja	Izbriši svu poštu s email servera
Preduvjet	Veza sa serverom je uspostavljena
Opis	
Korak	**Akcija**
1.	Korisnik poziva metodu za brisanje pošte s email servera
2.	Korisnik pokreće kod
3.	Email pošta je uspješno izbrisana
Extenzije	
Korak	**Akcija**
3a	Sandučić je prazan

Na tablici 4.10. naveden je slučaj korištenja koji opisuje funkcionalnost brisanja email pošte. Slučaj korištenja je napisan na temelju korisničke priče 2 iz tablice 4.2.

Tablica 4.11. Slučaj korištenja: Brisanje pošte po indeksu.

Slučaj korištenja	Izbriši email poruku s email servera prema navedenom indeksu poruke
Preduvjet	Veza sa serverom je uspostavljena
Opis	
Korak	**Akcija**
1.	Korisnik poziva metodu za brisanje pošte s email servera
2.	Korisnik navodi indeks i pokreće kod
3.	Email poruka je uspješno povučena u programski kod
Extenzije	
Korak	**Akcija**
2a	Index je negativan
4a	Sandučić je prazan

Na tablici 4.11. naveden je slučaj korištenja koji opisuje funkcionalnost brisanja email pošte. Slučaj korištenja je napisan na temelju korisničke priče pod rednim brojem 2. iz tablice 4.2.

Tablica 4.12. Slučaj korištenja: Rukovanje greškama.

Slučaj korištenja	Rukovanje greškama
Preduvjet	Neka od metoda je pozvana
Opis	
Korak	**Akcija**
1.	Korisnik poziva neku od metoda
2.	Unutar metode se događa greška
3.	Priključak rukuje greškom
4.	Priključak ispisuje grešku na konzolu i na grafičko sučelje

U tablici 4.12. naveden je slučaj korištenja koji opisuje rukovanje greškama. Slučaj korištenja je napisan na temelju korisničke priče 4 iz tablice 4.2. i korisničke priče 1 iz tablice 4.3.

Tablica 4.13. Slučaj korištenja: Testna automacija.

Slučaj korištenja	Testna automacija
Preduvjet	Funkcionalnosti su implementirane
Opis	
Korak	**Akcija**
1.	Razvojni inženjer razvija testove za svaku od funkcionalnosti
2.	Razvojni inženjer pokreće testove
3.	Testovi prolaze
Extenzije	
Korak	**Akcija**
3a	Testovi ne prolaze zbog loše implementiranih funkcionalnosti
3b	Testovi ne prolaze zbog loše implementiranih testova

Na tablici 4.13. naveden je slučaj korištenja koji opisuje potrebu za testnom automacijom. Slučaj korištenja je napisan na temelju korisničke priče 2 iz tablice 4.3.

4.2.2. Nefunkcionalni zahtjevi

U ovom poglavlju su definirani nefunkcionalni zahtjevi na temelju korisničkih priča iz poglavlja 4.1. Kod razvoja programskih rješenja nefunkcionalni zahtjevi su oni zahtjevi koji opisuju ne

ono što program radi ili treba raditi, već način na koji to treba izvršavati npr. zahtjevi na performanse, ograničenja dizajna, atributi kvalitete i sl. [20].

Kompatibilnost

Priključak mora biti kompatibilan s najnovijom verzijom cypressa (13.8.0). Zahtjev je napisan na temelju korisničke priče 7 iz tablice 4.2.

Minimalizacija ovisnosti

Priključak treba imati što manji broj ovisnosti, tj. ne smije sadržavati nepotrebne ovisnosti. Smanjenje broja ovisnosti olakšava održavanje, smanjuje rizik od konflikata među verzijama priključka te poboljšava sigurnost i performanse. Sve ovisnosti moraju biti dokumentirane i opravdane. Zahtjev je napisan na temelju korisničke priče 8 iz tablice 4.2.

Proširivost

Priključak mora imati jednostavan način za dodavanje novih funkcionalnosti i popratnih testova. Zahtjev je napisan na temelju korisničke priče 4 iz tablice 4.3.

Konvencija imenovanja

Programski kod priključka mora pratiti neku od poznatih konvencija imenovanja te imena moraju biti opisna i lako razumljiva. Zahtjev je napisan na temelju korisničke priče 3 iz tablice 4.3.

Performanse

Spajanje na email server mora biti izvršeno u vremenu kraćem od 2 sekunde. Povlačenje pošte iz sandučića mora biti izvršeno u vremenu kraćem od 1 sekunde kada sandučić sadrži manje od 10 email poruka. Brisanje pošte iz sandučića mora biti izvršeno u vremenu kraćem od 1 sekunde kada sandučić sadrži manje od 10 email poruka. Ovaj zahtjev se temelji na korisničkoj priči 6 iz tablice 4.2.

Dokumentacija

Funkcionalnosti priključka moraju biti dokumentirane unutar *README.md* datoteke zajedno s primjerima korištenja. Zahtjev je napisan na temelju korisničke priče 9 iz tablice 4.2.

Koristivost

Programsko sučelje priključka mora biti jednostavno i lako razumljivo. Zahtjev je napisan na temelju korisničke priče 3 iz tablice 4.2.

Sigurnost

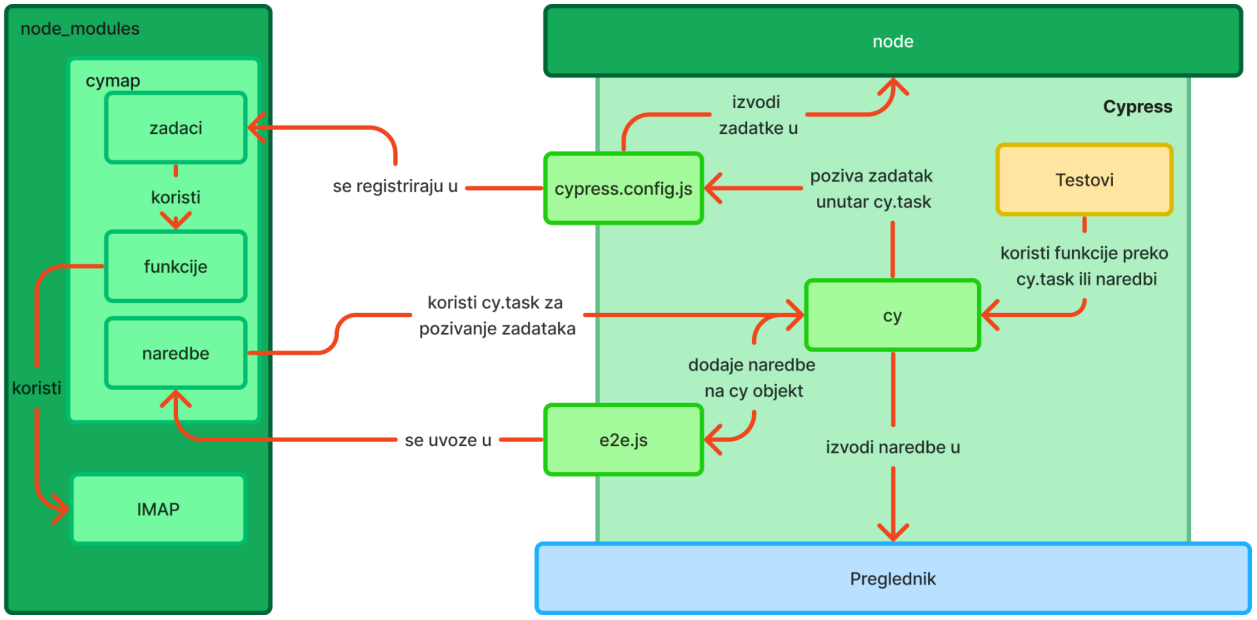
Prijenos podataka od servera do klijenta treba biti zaštićen. Zahtjev nije tražen unutar korisničkih priča, no sigurnost je implicitna kod klijent server komunikacije i ako je moguće uvijek treba biti implementirana.

4.3. Dizajn programskog rješenja

Dizajn programskog rješenja predstavlja planiranje izgleda i funkcioniranja programskog rješenja na temelju definiranih zahtjeva. U ovom poglavlju prikazana je arhitektura programskog rješenja te dizajn korisničkog sučelja

4.3.1. Arhitektura

Arhitektura u kontekstu računarstva predstavlja širok pojam i ima mnogo značenja i definicija. Za potrebe ovog rada arhitektura je definirana kao apstraktna struktura sustava prikazana komponentama sustava i međusobnim odnosima tih komponenti [21]. Arhitektura je dizajnirana na temelju funkcionalnih zahtjeva i cypress dokumentacije.



Slika 4.1. Dijagram komponenti

Na slici 4.1. prikazan je dijagram koji prikazuje način na koji priključak funkcionira s cypressom. Priključak je pohranjen u standardnom *node_modules* direktoriju koji je generiran pri inicijaliziranju projekta. Priključak sadrži funkcije, naredbe i zadatke. Funkcije su

implementacije željenih funkcionalnosti pomoću IMAP biblioteke. Implementirane funkcije je potrebno uvesti u zadatke gdje se funkcije slažu u objekt koji se vraća kao modul. Taj modul se uvozi u *cypress.confgi.js* i dodaje u *on* metodu na *task* događaj. Naredbe su pozivi *Cypress.Commands.add* metode koje pozivaju zadatke preko *cy.task* metode (Slika 3.5.) i uvoze se unutar *e2e.js* standardne cypress datoteke koja se izvršava pri pokretanju i time dodaje sve korisničke naredbe. Korisnik može koristiti metode preko *cy.task*, no ideja je da se koriste preko korisničkih naredbi zbog čitljivosti. Ideja iza ovakvog pristupa je da korisnik priključka nakon što instalira priključak može koristiti naredbe preko cypressovog sučelja unutar testova kao što je prikazano na slici 4.1. gdje se vidi da nakon što su zadaci registrirani i naredbe uvedene u *e2e.js* jedina stvar koji korisnik koristi je i dalje *cy*. Ovakav pristup pruža jednostavnost korištenja i brzu instalaciju.

4.3.2. Dizajn korisničkog sučelja

Korisničko sučelje priključka predstavlja skup funkcija koje korisnik koristiti kod implementacije priključka u vlastiti projekt. U ovom poglavlju su definirane te funkcije s njihovim ulazima i izlazima. Korisničko sučelje je dizajnirano na temelju funkcionalnih zahtjeva i arhitekture.

```
declare namespace Cypress {
    interface Chainable {

     setConnectionConfig(config:any):Chainable<null>

     getAllMail():Chainable<Array<Object>>

     deleteAllMail():Chainable<bool>

     getEmailByIndex(index:Number):Chainable<Object>

     deleteEmailByIndex(index:Number=1):Chainable<Object>
  }
}
```

Slika 4.2.Korisničko sučelje.

Na slici 4.3. prikazan je typescript deklaracijski dokument koji služi za definiranje tipova za dodatne korisničke naredbe [22]. U ovom poglavlju služi kao prikaz dizajna korisničkog sučelja. Korisničko sučelje se sastoji od metode za uspostavljanje veze te metoda za povlačenje i brisanje pošte iz sandučića.

```
{
  password:"socadscjfa",
  user:"some.gmail@gmail.com",
  host:'imap.gmail.com',
  port:993,
  tls:true,
  tlsOptions: { rejectUnauthorized: false }
}
```

Slika 4.3. Konfiguracijski objekt za IMAP.

Na slici 4.3. prikazan je konfiguracijski objekt za IMAP, to je objekt koji potrebno proslijediti metodi za postavljanje veze (*setConnectionConfig(config)*) sa slike 4.2.


```
{
    "attachments": [],
    "headers": {},
    "headerLines": [...],
    "html": "<p>Hello World </p>",
    "text": "Hello World",
    "textAsHtml": "<p>Hello World</p>",
    "subject": "Hello World",
    "date": "2024-04-30T16:15:16.000Z",
    "to": {
        "value": [
            {
                "address": "some.gmail@gmail.com",
                "name": ""
            }
        ],
        "html": "<span class=\"mp_address_group\"><a
href=\"mailto:some.gmail@gmail.com\"
class=\"mp_address_email\">some.gmail@gmail.com.com</a></span>",
        "text": "some.gmail@gmail.com"
    },
    "from": {
        "value": [
            {
                "address": "some.gmail@gmail.com",
                "name": ""
            }
        ],
        "html": "<span class=\"mp_address_group\"><a
href=\"mailto:some.gmail@gmail.com\"
class=\"mp_address_email\">some.gmail@gmail.com</a></span>",
        "text": "some.gmail@gmail.com"
    },
    "messageId": "<sdf@gmail.com>"
}
```


Slika 4.4. JSON struktura email poruke.

Na slici 4.3. prikazana je JSON struktura email objekta koji je vraćen pozivom metoda za povlačenje pošte.

4.4. Implementacija

Na temelju prethodno definiranih zahtjeva i dizajna u ovom poglavlju prikazana je implementacija priključka.

4.4.1. Odabir tehnologija

Kod tehnologija u *npm* okruženju potrebno je razlikovati ovisnosti (engl. *dependencies*) i ovisnosti za razvoj (engl. *development dependencies, devDependencies*). Ovisnosti su tehnologije koje su potrebne priključku za funkcioniranje nakon što je dodan u projekt. Ovisnosti za razvoj su tehnologije koje se koriste za razvoj priključka i nisu potrebne unutar direktorija nakon što je priključak dodan. Dobar primjer ovisnosti za razvoj je cypress jer cypress nije potreban za korištenje proizvoda, već se koristi tijekom razvoja za verifikaciju i validaciju funkcionalnosti.

Tablica 4.14. Ovisnosti.

Tehnologija	Opis
Node-imap[17]	Priključak koji omogućuje jednostavnu interakciju s email serverima s operacijama poput dohvaćanja, pretraživanja i upravljanja e-poruka
Mailparser[18]	Priključak koji omogućava parsiranje podataka koji su dobiveni sa servera u podatke razumljive ljudima

U tablici 4.14. su navedene tehnologije koje priključak koristi za izvršavanje svojih funkcija.

Tablica 4.15. Ovisnosti u razvoju.

Tehnologija	Opis
Cypress	Alat za testiranje priključka
Dotenv[19]	Alat za upravljanje varijablama okruženja
Nodemailer[20]	Alat za simuliranje slanja pošte unutar testova

U tablici 4.15. su navedene tehnologije koje su korištene pri razvoju priključka.

[17] Node-imap, https://www.npmjs.com/package/node-imap, 21.06.2024.
[18] Mailparser, https://www.npmjs.com/package/mailparser, 21.06.2024.
[19] Dotenv, https://www.npmjs.com/package/dotenv, 21.06.2024.
[20] Nodemailer, https://nodemailer.com/, 21.06.2024.

4.4.2. Funkcije

U ovom poglavlju je pokazana jedna od funkcija iz priključka kako bi se objasnilo na koji način priključak funkcionira. Na slici 4.5. prikazana je funkcija *deleteAllMail* koja uspostavlja IMAP vezu, otvara inbox, pretražuje sve poruke, označava ih kao obrisane i zatvara inbox. Ako se dogodi greška tijekom bilo kojeg koraka, funkcija vraća odbijeni *Promise* s greškom ili vraća uspješno ispunjen *Promise*. Funkcija za brisanje pošte po indeksu označuje samo poruku s primljenim indeksom. Funkcija za povlačenje email poruka radi na sličan način osim što ona umjesto označivanja svake poruke povlači svaku poruku, parsira ju, sprema ju u listu i vraća tu listu.

```
function deleteAllMail(){
    return new Promise((resolve, reject) => {
        ImapProxy.createConnection()
        const imap = ImapProxy.getConnectionInstance()
        imap.once("ready", ()=>{
            imap.openBox("INBOX", false,(err, box)=>{
                if (err) {
                    console.log(err)
                    return reject(err)
                }
                if(box.messages.total===0)return resolve(true)
                console.log(box)
                imap.search(["ALL"], (err, results)=>{
                    if (err) return reject(err)
                    console.log(results)
                    for (const uid of results) {
                        console.log(uid)
                        imap.addFlags(uid, "Deleted")
                    }
                    imap.closeBox((err)=>{
                        console.log(err)
                        if (err) return reject(err)
                        imap.end()
                        resolve(true)
                    })
                })
            })
        })
        imap.once('error', (err)=>{
            console.log(err)
            if (err) return reject(modifyError(err))
        });
        imap.once("end", (err)=>{
            console.log(err)
            if (err) return reject(err)
        })
        imap.connect()

}
```

Slika 4.5. Funkcija za brisanje email poruke.

4.4.3. Zadaci

Na slici 4.6. prikazan je modul koji vraća objekt sa svim metodama koje je potrebno registrirati. Ovaj modul se uvozi u *cypress.config.js* i referencira u *setupNodeEvents*.

```
const ImapProxy = require("../src/ImapProxy")
const {getAllMail, deleteAllMail, getEmailByIndex,
deleteEmailByIndex} = require("./commands")

module.exports={
    setConnectionConfig(config){
      ImapProxy.setConnectionConfig(config)
      return null
    },
    async getAllMail(){
      const mail = await getAllMail()
      return mail
    },
    async deleteAllMail(){
      const areDeleted = await deleteAllMail()
      return areDeleted
    },
    async getEmailByIndex(index){
      const email = await getEmailByIndex(index)
      return email
    },
    async deleteEmailByIndex(index){
      const isDeleted = await deleteEmailByIndex(index)
      return isDeleted
    }
}
```

Slika 4.6. Zadaci koji se uvoze u *cypress.config.js.*

Modul na slici 4.6. sadrži zadatke sa slike 4.1. Kao što je objašnjeno u poglavlju 4.3.1. zadaci uvoze funkcije i stavljaju ih u objekt koji se izvozi kao modul te se uvozi u *cypress.config.js.* Ovaj modul služi kao sučelje tj. moguće je definirati sasvim novu funkciju za brisanje pošte i jedina stvar koju je potrebno napraviti je uvesti ju u zadatke i pozvati ju u prikladnoj metodi te izbrisati poziv stare funkcije.

4.4.4. Omotači

Na slici 4.7. prikazan je modul koji sadrži omotače za zadatke iz poglavlja 4.4.3. Datoteka sa slike 4.7. se uvozi u standardnu *e2e.js* datoteku koja se izvodi pri pokretanju cypresa te dodaje korisničke naredbe. Osim metoda predviđenih dizajnom sadrži i *pasteHtml* koja služi kao pomoćna (engl. *utility*) metoda.

```
Cypress.Commands.add("pasteHtml", (html)=>{
    cy.document().invoke('write', html);
})

Cypress.Commands.add("setConnectionConfig", (config)=>{
    cy.task("setConnectionConfig", config)
})

Cypress.Commands.add("getAllMail", ()=>{
    return cy.task("getAllMail")
})

Cypress.Commands.add("deleteAllMail", ()=>{
    return cy.task("deleteAllMail")
})

Cypress.Commands.add("getEmailByIndex", (index=1)=>{
    return cy.task("getEmailByIndex", index)
})

Cypress.Commands.add("deleteEmailByIndex", (index=1)=>{
    return cy.task("deleteEmailByIndex", index)
})
```

Slika 4.7. Funkcija za brisanje email poruke.

4.5. Korištenje

Na slici 4.8. prikazan je cypress test koji postavlja IMAP konfiguraciju, prijavljuje korisnika i provjerava primanje email poruka s naslovom "Autentikacija". Nakon prijave, test dohvaća sve email poruke, provjerava prvu email poruku za tekst "Ulogirani ste", a zatim briše sve email poruke. Test osigurava da se autentifikacijska email poruke uspješno šalje i prima.

```
describe('tests authentication email function', () => {
    before("set configs", () => {
        cy.setConnectionConfig(Cypress.env("imapConfig"))
    })

    it('sends authentication email', () => {
        cy.login(Cypress.env("username"),
Cypress.env("password")).then(() =>
            cy.getAllMail().then(mail => {
                expect(mail[0].subject).to.contain("Autentikacija")
                cy.pasteHtml(mail[0].html)
                cy.contains("Ulogirani ste.")
                cy.deleteAllMail()
            })
        )
    })
})
```

Slika 4.8. Funkcija za brisanje email poruke.

4.6. Automatski testovi

Automatski testovi osiguravaju kvalitetu softvera kroz detekciju grešaka prije objave priključka. Potrebno je imati konfigurirane testove koji pokrivaju što više slučajeva korištenja funkcionalnosti priključka kako bi se osigurao ispravan rad. Kod ovog pristupa, promjene u kodu mogu se brzo testirati i povratna informacija se dobiva odmah, što olakšava otkrivanje problema. Automatskim testovima se osigurava stabilnost i pouzdanost koda, a redovito pokretanje testova osigurava da nove promjene ne narušavaju postojeću funkcionalnost. Za priključak su napisana 22 testna slučaja (P.4.1.) te su ti testni slučajevi automatizirani u cypressu.

4.7. Objavljivanje

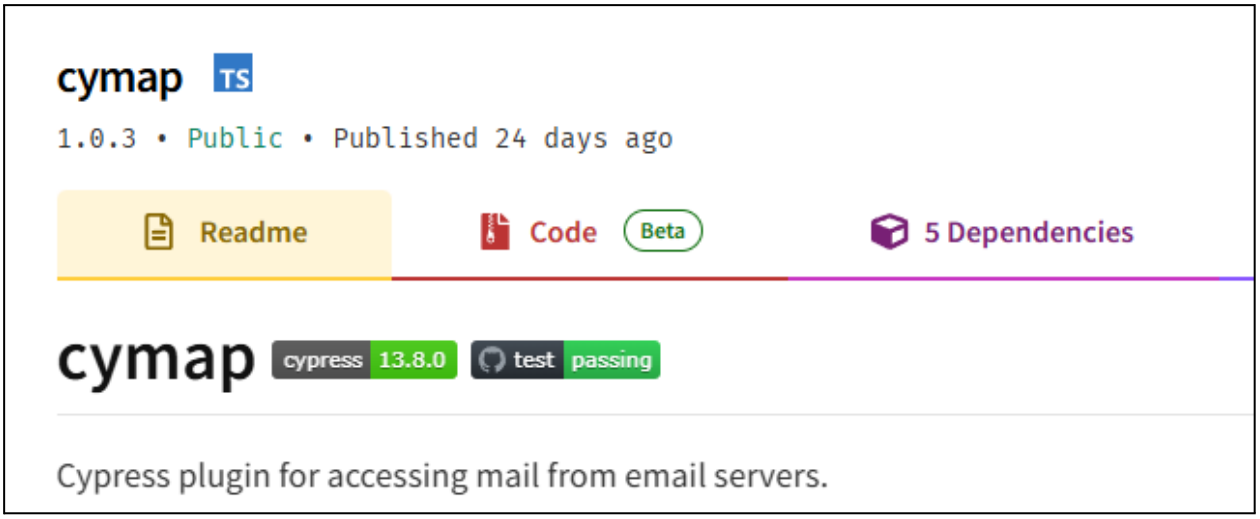
Kako bi se priključak objavio mora sadržavati *package.json,* kod i dokumentaciju (*README.md*). Potrebno je dodati i *.npmignore* i tamo navesti sve datoteke koje nije potrebno uvoziti pri instalaciji priključka. Potrebno je napraviti *npm* korisnički račun te se autentificirati u projektu s naredbom *npm login*. Nadalje je potrebno provjeriti je li ime priključka zauzeto, ako nije pokrenuti naredbu *npm publish.* Prethodno opisani niz akcija se za priključak izvodi automatski pomoću github akcija koje izvršavaju objavljivanje i testiranje priključka.

```
name: test
on: workflow_dispatch

jobs:
  test:
    name: Test
    runs-on: windows-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v3
        with:
          node-version: 20.11.1
      - run: npm ci
      - name: create env file
        run: |
          echo USER=${{ secrets.USER }} >> .env
          echo PASS=${{ secrets.PASS }} >> .env
      - run: npm test
```

```
name: publish
on: workflow_dispatch

jobs:
  publish:
    name: Publish
    runs-on: windows-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v3
        with:
          node-version: 20.11.1
          registry-url: https://registry.npmjs.org/
      - run: npm ci
      - run: npm publish
        env:
         NODE_AUTH_TOKEN: ${{secrets.NPM_TOKEN}}
```

Slika 4.9. Github akcije.

Na slici 4.9. su prikazane github akcije korištene za objavu i izvršavanje testova nad priključkom. Nakon što su akcije iznad uspješno izvršene priključak je objavljen i test bedž je generiran.



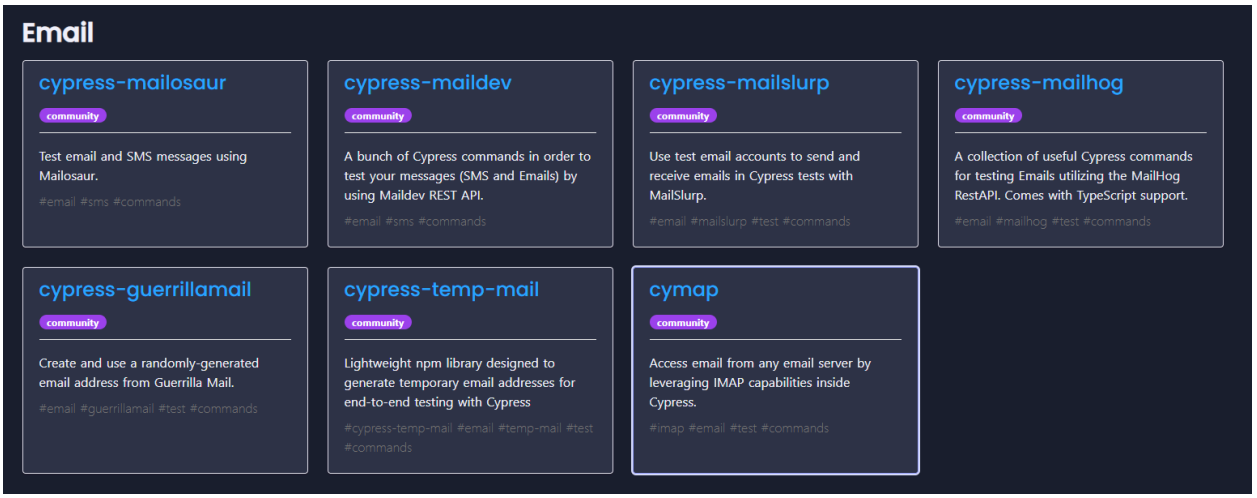
Slika 4.10. Cymap[21].

Na slici 4.10. prikazana je *npm* stranica priključka gdje se vidi test bedž zajedno s bedžom koji prikazuje cypress verziju koju priključak podržava.

4.7.1. Dodavanje priključka u cypress dokumentaciju

Dodavanje vlastitog priključka u dokumentaciju radi se podnošenjem zahtjeva za izmjene (engl. *pull request*, *merge request*) na repozitorij cypress dokumentacije[22]. Zahtjev[23] treba dodati podatke o priključku unutar *plugins.json* datoteke. Nakon što je zahtjev napravljen, osoba autorizirana za odobravanje zahtjeva treba pregledati priključak i zahtjev. Kako bi priključak bio dodan treba imati: jasno definiranu svrhu, jasnu dokumentaciju, integracijske testove s cypressom, CI (engl. continuous integration) pipu i kompatibilnost s najnovijom verzijom cypressa [23]. Nakon što je zahtjev za izmjene odobren priključak je dodan s ostalim email priključcima napravljenim od zajednice.

[21] Cymap, https://www.npmjs.com/package/cymap, 23.06.2024.
[22] Github repozitorij cypress dokumentacije, https://github.com/cypress-io/cypress-documentation, 23.06.2024.
[23] Zahtjev za izmjene nad cypress dokumentacijom, https://github.com/cypress-io/cypress-documentation/pull/5839, 23.06.2024.



Slika 4.11. Izlistanje priključaka iz cypress dokumentacije.

Na slici 4.11. prikazani su priključci predloženi od cypressa za testiranje email servisa i među njima je priključak izrađen u sklopu ovog diplomskog rada.

5. USPOREDBA PREDLOŽENOG RJEŠENJA S POSTOJEĆIM

U sklopu rada je izrađen novi email priključak koji se u ovom poglavlju uspoređuje s postojećim priključcima. Istraživanje obuhvaća analizu karakteristika i funkcionalnosti kako bi se ocijenila korisnost i inovativnost novog priključka u odnosu na već postojeće opcije. Fokus je na detaljnom opisu prednosti novog priključka, kao i na istraživanju mogućnosti poboljšanja korisničkog iskustva i ostalih osobnosti u usporedbi s trenutnim rješenjima. Priključci su bodovani po sljedećim parametrima: broju funkcionalnosti, broju automatskih testova, broju ovisnosti, činjenicom plaća li se i činjenicom je li ovisi o REST ill programskom sučelju servisa treće strane.

Broj funkcionalnosti predstavlja ključni parametar za procjenu vrijednosti novog email priključka u usporedbi s postojećim rješenjima. Više funkcionalnosti može poboljšati korisničko iskustvo i povećati efikasnost, ali istovremeno može zahtijevati bolju podršku i veću kompleksnost u implementaciji.

Broj automatskih testova je parametar kojime se može procijeniti kvaliteta priključka u usporedbi s postojećim rješenjima. Veći broj automatskih testova može značajno povećati pouzdanost i stabilnost softvera jer omogućava brže otkrivanje i ispravljanje grešaka.

Pod ovisnost misli se na ovisnosti iz tablice 4.14. Veći broj ovisnosti može ubrzati razvoj korištenjem postojećih biblioteka i alata, ali također može povećati složenost sustava, rizik od kompatibilnosti, sigurnosnih problema i otežanog održavanja. Svaka dodatna ovisnost može zahtijevati dodatne resurse za praćenje i ažuriranje, smanjujući dugoročno agilnost razvoja.

Kad se priča o plaćanju ne misli se na direktno plaćanje za korištenje priključka. Svi priključci su otvorenog koda. Misli se na treći servis na kojemu je potrebno kupiti ključ za korištenje. Besplatno rješenje može značajno smanjiti troškove implementacije i održavanja, što ga čini privlačnim za organizacije s ograničenim budžetom. Međutim, plaćeno rješenje često dolazi s boljom podrškom, redovitim ažuriranjima i dodatnim funkcionalnostima koje mogu poboljšati učinkovitost i sigurnost.

Činjenica je li priključak ovisi o REST sučelju ili programskom sučelju servisa treće strane predstavlja parametar za procjenu fleksibilnosti priključka. Ako email server s kojim priključak komunicira tijekom rada prestane funkcionirati nije moguće nastaviti s izvođenjem testova dok se server ne vrati u funkcionalno stanje. Također ako programsko sučelje servisa prestane raditi ispravno automatski testovi neće funkcionirati.

Tablica 4.15. Karakteristike priključaka.

Priključak	**Funkcionalnosti**	**Automatski testovi**	**Broj ovisnosti**	**Plaća se**	**Ovisnosti o serveru/servisu**
Cypress-mailosaur	32	39	0	DA	DA
Cypress-maildev	10	16	1	NE	DA
Cypress-mailslurp	38 klasa	54	1	DA	DA
Cypress-mailhog	20	52	0	NE	DA
Cypress-guerillamail	9	5	0	NE	DA
Cypress-temp-mail	2	1	3	NE	DA
cymap	5	22	2	NE	NE

Na temelju podataka u tablici 4.15. može se zaključiti da je cymap priključak koji, iako ima nižu razinu funkcionalnosti (5) u usporedbi s ostalim priključcima poput cypress-mailslurpa (38 klasa) i cypress-mailosaur (32), ima nekoliko ključnih prednosti koje ga čine atraktivnim rješenjem u određenim scenarijima. Manji broj funkcionalnosti može biti prednost za timove i projekte koji trebaju jednostavno i specifično rješenje te ga može biti lakše razumjeti, implementirati i održavati, smanjujući vrijeme potrebno za obuku i integraciju u postojeći sustav.

S 22 automatska testa, cymap nudi dobru razinu pokrivenosti za osnovne funkcionalnosti, osiguravajući da osnovne operacije rade ispravno, što može biti dovoljno za projekte koji ne zahtijevaju napredne funkcionalnosti. Osim toga, dobro razvijeni automatski testovi mogu pomoći u brzom otkrivanju problema i osiguravanju stabilnosti tijekom razvoja.

Iako cymap ima dvije ovisnosti, ovaj broj je razuman i može se lako upravljati. U usporedbi s cypress-mailslurpom i cypress-maildevom koji imaju po jednu ovisnost, cymap je možda malo složeniji, ali je i dalje upravljiv. Ovisnosti su često potrebne za proširenje funkcionalnosti i poboljšanje performansi, a cymap s dvije ovisnosti sugerira da su ovisnosti pažljivo odabrane kako bi se uravnotežila funkcionalnost i složenost.

Cymap je besplatan, što ga čini atraktivnim izborom za projekte s ograničenim budžetom. Usporedivši to s plaćenim opcijama poput cypress-mailslurpa i cypress-mailosaura, cymap omogućuje timovima da alociraju sredstva na druge kritične dijelove projekta, poput razvoja novih značajki ili marketinga.

Nedostatak ovisnosti o serveru ili servisu, za razliku od ostalih priključaka poput cypress-mailhog-a ili cypress-guerillamail-a, znači da je cymap fleksibilniji i može se lakše prilagoditi različitim okruženjima. Ova karakteristika omogućava lakšu integraciju s različitim sustavima i smanjuje potrebu za prilagođavanjima pri svakoj promjeni na serveru ili servisu. Fleksibilnost koju pruža cymap čini ga pogodnim za projekte koji zahtijevaju česte promjene ili rade u dinamičnim okruženjima. Osim toga, postoji mogućnost dodavanja još funkcionalnosti u budućnosti, čime se može dodatno poboljšati njegova upotrebljivost i prilagodljivost različitim potrebama.

6. ZAKLJUČAK

U ovom diplomskom radu ukratko je objašnjeno stanje email tehnologije te trenutni pristupi testiranja neovisni o tehnologiji, s primjerima korištenja. Fokus rada je na automatiziranom testiranju email servisa pomoću cypress razvojnog okruženja. Objašnjeno je što je email i kako ga testirati, te su detaljno prikazani postojeći pristupi testiranju email servisa unutar cypressa.

Središnji dio rada donosi prijedlog novog priključka za testiranje email servisa, osmišljenog da nadomjesti nedostatke postojećih rješenja i omogući besplatno testiranje email servisa. U tom dijelu rada, detaljno su opisani svi aspekti razvoja programskih rješenja, uključujući definiranje korisničkih zahtjeva i specifikacija programskog rješenja kroz razne formalne i neformalne metode kao što su korisničke priče, slučajevi korištenja, dijagrami komponenti i definicija nefunkcionalnih zahtjeva. Nadalje, definirani je i objašnjen dizajn programskog rješenja i prikazana je implementacija osnovnih dijelova.

Na kraju, prikazano je korištenje, automatski testovi i objava priključka, te je priključak uspoređen s postojećim rješenjima. Ovim radom ostvaren je doprinos cypress zajednici kroz objavu priključka i njegovo integriranje s cypress dokumentacijom putem zahtjeva za dodavanje u repozitorij. Priključak je pregledan od strane autorizirane osobe iz Cypress.io i prihvaćen kao jedan od priključaka zajednice u njihovoj dokumentaciji gdje ga se trenutno može pronaći.

Ovaj rad doprinosi razumijevanju i unapređenju procesa testiranja email servisa, pružajući vrijedne uvide i praktične alate za razvojne inženjere. Iako priključak trenutno sadrži minimalan broj funkcionalnosti za uspješno automatiziranje testova email servisa, potrebno je proširiti broj funkcionalnosti i napisati odgovarajući broj testova za svaku funkcionalnost kako bi se osigurala kvaliteta programskog rješenja.

LITERATURA

[1] N., S.Baron, *Alphabet to Email*. Routledge: New York, 2000.
[2] V., V. Raibov, *SMTP*. Rivier College, 2005.
[3] D., Mullet, K., Mullet, *Managing IMAP*, First. O'REILLY, 2000.
[4] A., Shahbaz, „Performance Evaluation of IMAP and POP3 Protocols Using Optimized Network Engineering Tool (OPNET)", Master, University of Sindh, Pakistan, 2014.
[5] G., Bahmutov, „Full Testing of HTML Emails using SendGrid and Ethereal Accounts" [online], 24-svi-2021. Dostupno na: https://www.cypress.io/blog/2021/05/24/full-testing-of-html-emails-using-ethereal-accounts . [Pristupljeno: 29.5.2024.].
[6] G., Bahmutov, „Testing HTML Emails using Cypress" [online], 11-svi-2021. Dostupno na: https://www.cypress.io/blog/2021/05/11/testing-html-emails-using-cypress. [Pristupljeno: 30.5.2024.].
[7] H., Ben Tayeb, „Mailcow: Setting up a full featured self hosted mail server" [online], 30-svi-2024. Dostupno na: https://dev.to/hatembentayeb/mailcow-setting-up-a-full-featured-self-hosted-mail-server-4511. [Pristupljeno: 30.5.2024.].
[8] C., Le Biez, „Cypress Maildev" [online], 16-velj-2023. Dostupno na: https://github.com/Clebiez/cypress-maildev?tab=readme-ov-file#cypress-maildev. [Pristupljeno: 5.6.2024.].
[9] D., Shcherbakan, „Send Emails with Gmail API" [online], 11-stu-2023. Dostupno na: https://mailtrap.io/blog/send-emails-with-gmail-api/. [Pristupljeno: 30.5.2024.].
[10] Mailosaur, „Mailosaur Cypress Commands" [online], 05-srp-2022. Dostupno na: https://github.com/mailosaur/cypress-mailosaur?tab=readme-ov-file#mailosaur-cypress-commands. [Pristupljeno: 5.6.2024.].
[11] L., da Costa, *Testing JavaScript Applications*. Manning Publications Co.: New York, 2021.
[12] Cypress.io, „Cypress Architecture" [online], 2024. Dostupno na: https://docs.cypress.io/guides/overview/key-differences#Architecture. [Pristupljeno: 3.6.2024.].
[13] Cypress.io, „Cypress Debugability" [online], 2024. Dostupno na: https://docs.cypress.io/guides/overview/key-differences#Debuggability. [Pristupljeno: 3.6.2024.].
[14] Cypress.io, „Cypress Trade Offs" [online], 2024. Dostupno na: https://docs.cypress.io/guides/references/trade-offs. [Pristupljeno: 3.6.2024.].
[15] Cypress.io, „Cypress custom commands" [online], 2024. Dostupno na: https://docs.cypress.io/api/cypress-api/custom-commands. [Pristupljeno: 3.6.2024.].
[16] I., Kent, „MailHog" [online], 02-kol-2022. Dostupno na: https://github.com/mailhog/MailHog?tab=readme-ov-file#mailhog-----. [Pristupljeno: 5.6.2024.].
[17] QAT, „Guide to User Requirements" [online], 27-ožu-2024. Dostupno na: https://qat.com/guide-user-requirements/. [Pristupljeno: 12.6.2024.].
[18] A., Cockburn, *Writing effective use cases*. .
[19] K., Wiegers, J., Beatty, *Software Requirements*, 3. izd. Microsoft Press: U.S., 2013.
[20] L., Chung, B., A. Nixon, E., Yu, J., Mylopoulos, *Non-functional requirements in software engineering*. Springer.
[21] I., Gorton, *Essential Software Architecture*. Springer: New York.
[22] Cypress.io, „Types for Custom Commands" [online], 2024. Dostupno na: https://docs.cypress.io/guides/tooling/typescript-support#Types-for-Custom-Commands.
[23] Cypress.io, „Adding plugins" [online]. Dostupno na:

https://github.com/cypress-io/cypress-documentation/blob/main/CONTRIBUTING.md#adding-plugins.

ŽIVOTOPIS

Filip Cica, rođen je 13. lipnja 2000. u Novog Gradišci. Sveučilišni preddiplomski studij Računarstvo završava 2022. godine i upisuje diplomski sveučilišni studij Računarstvo, smjer Programsko inženjerstvo u sklopu Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek. Tijekom svog obrazovanja aktivno volontira u raznim udrugama i organizacijama. Tijekom diplomskog studija surađuje s ICT tvrtkom Barrage kao razvojni inženjer softvera u testu i radi kao predavač instrukcija u tvrtci Edukos. Tijekom svoje zadnje godine sudjelovao je u FERIT Pro Student projektu na temelju kojeg prima priznanje za nagrađenu poduzetničku ideju. Nadalje, sudjelovao je u *Start For Future* projektu kao dio tima koji osvaja prvo mjesto za poduzetničku ideju. Također sudjeluje i u natječaju Digitalnog Inovacijskog Inkubatora kao dio tima koji je proglašen kao generalni pobjednik natječaja.

SAŽETAK

Ovaj diplomski rad istražuje email tehnologiju i pristupe njezinom testiranju, s posebnim naglaskom na cypress razvojno okruženje. Objašnjava koncept email tehnologije i metode za njezino testiranje unutar cypressa. Centralni dio rada predstavlja novi priključak za testiranje email servisa, razvijen s ciljem otklanjanja nedostataka postojećih rješenja i omogućavanja besplatnog testiranja. Priključak je integriran u cypress dokumentaciju nakon pregleda od strane razvojnih inženjera iz Cypress.io. Rad naglašava važnost automatizacije testiranja email servisa zbog njegove kompleksnosti i učestalih potreba za raznim scenarijima testiranja. Predloženo rješenje koristi IMAP protokol za interakciju s email serverom, omogućujući testiranje prema različitim email serverima uključujući popularne poput Gmaila. Ovim radom se unapređuje razumijevanje i primjena procesa testiranja email servisa, pružajući korisne alate za inženjere u razvoju softvera.

Ključne riječi: cypress, email servis, email server, IMAP, automatsko testiranje

ABSTRACT

Automated email service testing using the Cypress framework

This thesis explores email technology and approaches to its testing, focusing on the Cypress development environment. It explains the concept of email and methods for testing it within cypress. The central part of the thesis introduces a new plugin for testing email services, developed to overcome shortcomings of existing solutions and enable free testing. The plugin is integrated into cypress documentation after review by Cypress.io. The thesis emphasizes the importance of automating email service testing due to its complexity and frequent need for various testing scenarios. The proposed solution uses the IMAP protocol for interacting with email servers, allowing testing against different servers including popular ones like Gmail. This thesis enhances understanding and application of email service testing processes, providing valuable tools for software development engineers.

Keywords: cypress, testing, email service, email server, IMAP, automated testing

P.4.1.

Tablica 1. Testni slučajevi

r.b.	Naziv testa	Opis testa	Očekivani rezultat	Funkcija
1	Vraća *true* nakon brisanja email pošte	Briše sve email poruke i provjerava vraća li funkcija *deleteAllMail* true	*deleteAllMail* vraća true	*deleteAllMail*
2	Vraća *true* ako je sandučić prazan pri brisanju	Briše sve email poruke i ponovno briše kako bi provjerio vraća li funkcija *deleteAllMail* true kada je sandučić prazan	*deleteAllMail* vraća true kada je sandučić prazan	
3	*getAllMail vraća* 0 nakon pozivanja *deleteAllMail*	Provjerava ima li email poruka, briše sve email poruke i provjerava vraća li *getAllMail* 0 email poruka nakon brisanja	*getAllMail* vraća 0 email poruka nakon što je pozvana funkcija *deleteAllMail*	
4	Briše email poruku po indeksu	Briše email poruku s određenim indeksom nakon što pošalje 4 email poruke	*deleteEmailByIndex* uspješno briše email poruku, a *getAllMail* vraća 3 email poruke	*deleteEmailByIndex*
5	Vraća grešku ako email poruka nije pronađena	Pokušava obrisati email poruku kada je sandučić prazan	Pojavljuje se greška s porukom "*Can't delete message, inbox empty.*"	
6	Vraća grešku ako je indeks izvan granica	Pokušava obrisati email poruku s indeksom izvan granica nakon slanja 4 e-poruka	Pojavljuje se greška s porukom "*Can't find the message.*"	
7	Vraća grešku ako je indeks negativan	Pokušava obrisati email poruku s negativnim indeksom nakon slanja 4 email poruka	Pojavljuje se greška s porukom "*Negative indexes are not allowed.*"	
8	*getAllMail* vraća jednu manje nakon pozivanja *deleteEmailByIndex*	Provjerava ima li email poruka, briše email poruku po indeksu i provjerava vraća li *getAllMail* jednu manje email poruku	*getAllMail* vraća jednu manje email poruku nakon što je pozvana funkcija *deleteEmailByIndex*	
9	Vraća parsirano tijelo	Vraća 4 email poruke prethodno poslane i provjerava svojstva svake email poruke	*getAllMail* vraća 4 email poruke, svaka s odgovarajućim svojstvima	*getAllMail*
10	Vraća prazan niz kada nema email poruka	Briše sve email poruke i provjerava vraća li funkcija *getAllMail* prazan niz	*getAllMail* vraća prazan niz	
11	Dohvaća email poštu prema indeksu	Šalje 4 email poruke i dohvaća email poruku s indeksom 2	Email poruka ima očekivana svojstva prema sadržaju poslane email poruke	*getEmailByIndex*
12	Vraća grešku ako email poruka nije pronađena	Briše sve email poruke i pokušava dohvatiti email poruku	Pojavljuje se greška s porukom "*No emails in the mailbox.*"	
13	Vraća grešku ako je indeks negativan	Pokušava dohvatiti email poruku s negativnim indeksom	Pojavljuje se greška s porukom "*Index cannot be smaller than 1.*"	
14	Vraća grešku ako je indeks 0	Pokušava dohvatiti email poruku s indeksom 0	Pojavljuje se greška s porukom "*Index cannot be smaller than 1.*"	

15	Vraća grešku na pogrešne vjerodajnice	Postavlja pogrešne vjerodajnice i pokušava dohvatiti email poruku	Pojavljuje se greška s porukom "*Wrong username or password.*"	setConnectionConfigs
16	Vraća grešku na nevažeći *host*	Postavlja nevažeći host i pokušava dohvatiti e-poruku	Pojavljuje se greška s porukom "*Bad host.*"	
17	Vraća grešku na nevažeći *port*	Postavlja nevažeći port i pokušava dohvatiti e-poruku	Pojavljuje se greška s porukom "*Timeout, check port and tls.*"	
18	Vraća grešku na nevažeći TLS	Postavlja nevažeći TLS i pokušava dohvatiti e-poruku	Pojavljuje se greška s porukom "*Timeout, check port and tls.*"	
19	Vraća grešku na praznu lozinku	Postavlja praznu lozinku i pokušava dohvatiti e-poruku	Pojavljuje se greška s porukom "*Wrong password or username.*"	
20	Vraća grešku na prazno korisničko ime	Postavlja prazno korisničko ime i pokušava dohvatiti e-poruku	Pojavljuje se greška s porukom "*Wrong password or username.*"	
21	Vraća grešku na praznu lozinku i prazno korisničko ime	Postavlja prazno korisničko ime i lozinku te pokušava dohvatiti email poruku	Pojavljuje se greška s porukom"*Wrong password or username.*"	
22	Vraća grešku kada *setConnectionConfigs* nije korišten	Ne postavlja konfiguraciju i pokušava dohvatiti email poruku	Pojavljuje se greška s porukom "*Config object empty.*"	